Assignment (v1.1) | ROS Navigation & Search Challenge Report

(Karthi Narendrababu Geetha-s3835901, Prabhat Kumar Singh-s3833321, Varsha Chandrasekhar Bendre-s3831858)

1. **A full technical description of your ROS package dependencies and configuration including**:

a. All ROS packages that are used in your software

- MoveBaseGoal, MoveBaseAction [**move_base_msgs.msg**]
- OccupancyGrid, Odometry, Path [**nav_msgs.msg**]
- ObjectsStamped [**find_object_2d.msg**]
- LaserScan [**sensor_msgs.msg**]
- Marker, MarkerArray [**visualization_msgs.msg**]
- actionlib
- find_object_2d
- slam_gmapping
- move_base
- tf

b. All additional libraries (such as OpenCV) that are used in your software

- Os
- Numpy

c. Description of launch files required to launch Search & Navigation Challenge software

- **initiate.launch:**
  i. find_object_2d package to recognise the start marker.
  ii. Calling the initiate.py file to initiate the exploration after seeing the start marker.
  iii. We are also using two static_transform_publisher to get camera and laser scan feed.

- **explore.launch:**
  i. SLAM_gmapping will generate a map environment for the rosbot. static_transform_publisher from the tf package will be obtaining the pose of the robot relative to the starting point and the laser scanner pose relative to the robot. This will subscribe to the topics /tf and /scan. The subscribed data will be published to the /map topic.
  ii. explore_lite: It is used to perform the exploration. find_object_2d package, which started in the initiate.launch will get killed. Now, find_object_2d initiates in the explore.launch file to find and recognize the hazard markers, while exploring the unknown environment.
  iii. The rosbot will move according to the cost_map to find the optimal path.
  iv. Finding the hazard markers and publishing it to the /hazards as a type visualization_msgs.Marker.
  v. We are publishing the /map (type OccupancyGrid) and the /path (type Path) from the package Nav_msgs.

d. All custom configuration files and training models

i. [**buildtool_depend**] Catkin
ii. [**build_depend, build_export_depend, exec_depend**] roscpp, rospy, std_msgs, move_base_msgs, nav_msgs, sensor_msgs, find_object_2d, geometry_msgs, actionlib, tf.

2. A full description of any ROS nodes that you wrote. For each node this should include

Node1: initiate

Description of the node functionality: It will recognise the start marker once the start marker is recognised. This will launch the explore.launch file using os package of python.

Description of why the node is necessary: To start the rosbot to explore.

Citations:

[1]. @misc{labbe11findobject,
Author = {{Labb\'{e}, M.}},
Howpublished = {\url{http://introlab.github.io/find-object}},
Title = {{Find-Object}},
Year = 2011
}
[2]. @inproceedings {6556373,
title = {tf: The transform library},
booktitle = {Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on},
series = {Open-Source Software workshop},
year = {2013},
month={April},
pages={1-6},
author = {Foote, Tully},
doi={10.1109/TePRA.2013.6556373},
ISSN={2325-0526}
}

Node 2: explore

Description of the node functionality: Using explore node, we will be calling the exploration.py file. This python file will be handling the operations throughout the exploration and finally redirect the rosbot to the starting position.

Description of why the node is necessary

- To get the hazard marker id's using find_object_2d, with the help of /odom topic the current position of the robot will be obtained and with the help of /scan topic, the X-Axis of the laser will be obtained and the distance from the rosbot current pose will be added to the laser's X-axis in order to get the distance between the rosbot and the hazard marker. The marker coordinates will be published as the topic "/hazards".
- To launch **rviz**, a visualization tool for the rosbot which helps to visualize what rosbot is seeing while exploring the environment. This tool will subscribe to the ros topics such as map, path, hazards, scan and camera with the help of the configuration file "**challenge.rviz**".

- To find the current pose of the rosbot with odometry, store it in the PoseStampedArray, and publish it to the path as type nav_msgs.msg. To extract the origin from the occupancy grid message using slam_gmapping.
- map_server map_saver will be used to save the map after exploration.
- The variable origin has the starting point, where the rosbot will be returning after the exploration.
- To shut down the rospy.
- Exploring the unknown environment – Explore_lite package is using move_base to navigate through the environment. The yaml configuration files used in the move_base are Local_costmap, Global_cost_map, Trajectory path planning, Exploration (The current motion is determined with the local cost map and the longer-range trajectory with the global cost map).

Citations:

[1]. @masterthesis{Hörner2016,
author = {Jiří Hörner},
title = {Map-merging for multi-robot system},
address = {Prague},
year = {2016},
school = {Charles University in Prague, Faculty of Mathematics and Physics},
type = {Bachelor’s thesis},
URL = {https://is.cuni.cz/webapps/zzp/detail/174125/},
}

[2]. Path planning, Unknown environment exploration, Map navigation, SLAM navigation, accessed April 2020, <https://husarion.com/tutorials/>

3. **An evaluation of the performance of your software. This evaluation should include the performance of your software during the Search & Navigation Challenge demonstrations. You should also conduct independent evaluations.**

Demonstrations (Image 1 and Image 2):

a. Three hazards were identified but the id was not returned, because they were set to global variable.
b. The path was not returned, as it was not published.
c. The type for the “/hazards” was given as MarkerArray instead of Marker, hence it was not published on the map.
d. The first 2 runs did not yield good results and ended the exploration in the middle of the maze, due to the warning [Unable to get starting pose of robot, unable to create global plan] Since our transform latency is very less, the robot was unable to obtain the transform between the robot's base_link and sensor. Due to which the robot stopped moving and was unable to explore the whole environment.

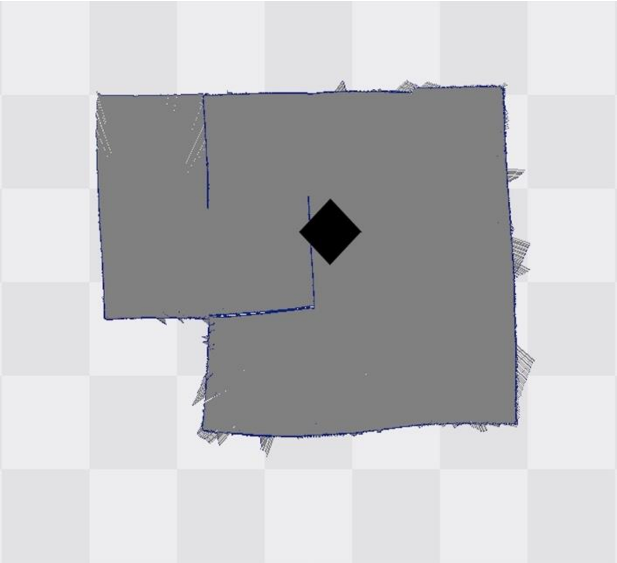

Image 1

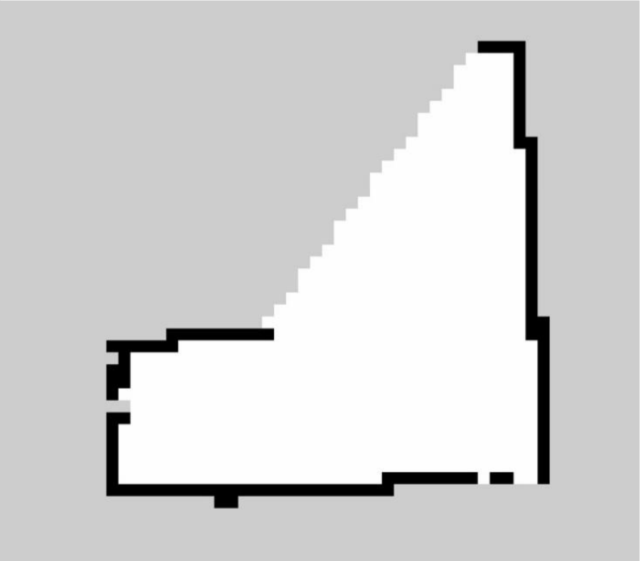

Image 2

Independent evaluations:

Strengths:

a. It was able to explore the map fully, as it was not too complex. (Image 3)
b. Identified the hazard markers instantly.
c. started the exploration instantly

Limitations:

a. The exploration took long time to end, thus it failed to go back in the final run. We checked the code separately and it was working properly, whenever the path was not very complicated.

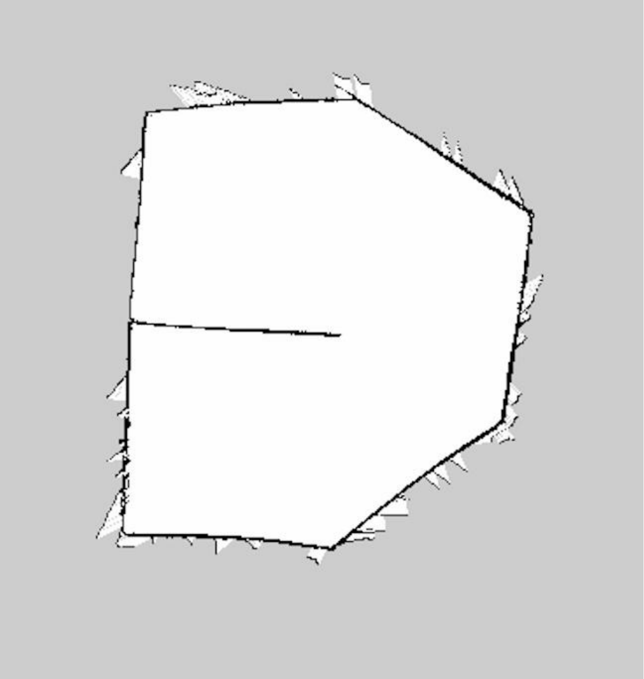

Image 3

4. **A critical analysis of the limitations of your software. You should specifically note any issues which may prevent your software completing some aspect of the Search & Navigation Challenge**

Strength:

a. It was recognizing the start marker and hazard markers instantly, and the exploration instantly.
b. We were able to explore the map smoothly, in the third run.

Limitations:

a. During the initial run, as the map was not complicated, it explored the whole map every time.
b. we invested time on subscribe_depth = true [which is based on find_object_3d], find_object_2d. At the end, we must use laser scan data to get the coordinates of hazard marker, we did a last- minute change to publish the hazards. We were publishing markers at the wrong frame_id.
c. while exploring, the rosbot was unable to find its own position, because of transform_tolerance as it was very less and because if this, rosbot stops the exploration, before completing the full map.
d. we are using global variables. To store the marker, which was getting overwritten every time, because of this, all the marker at the same coordinate